PATENT ABSTRACTS OF JAPAN

(11)Publication number : 06-168145
(43)Date of publication of application : 14.06.1994

(51)Int.Cl. G06F 9/46
G06F 12/00

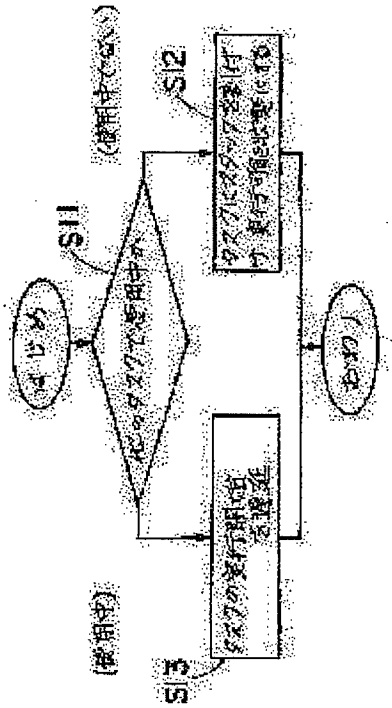
(21)Application number : 03-352297
(22)Date of filing : 13.12.1991

(71)Applicant : HITACHI LTD
(72)Inventor : SHIMIZU TAKESHI
TAKEYAMA HIROSHI
KOBAYAKAWA MANABU

(30)Priority
Priority number : 03 39269 Priority date : 08.02.1991 Priority country : JP

(54) STACK CONTROL SYSTEM AND MICROCOMPUTER

(57)Abstract:
PURPOSE: To provide a stack control system in a multitask processing system capable of easily performing processing to allocate a stack area to a task and improving memory working efficiency for the stack area.
CONSTITUTION: The same stack area is shared with plural tasks, and the allocation of the task to a shared stack area is controlled exclusively setting the execution of the task from start to completion as unit for individual shared task. For example, when the execution of the task is requested, it is decided whether or not another task which shares the stack area is using the shared task area (S11), and the task can be executed when another task is not being executed (S12), and transition to a state where the task can be executed is delayed when another task is being executed (S13).



LEGAL STATUS

[Date of request for examination] 11.12.1998
[Date of sending the examiner's decision of rejection]
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number] 3212656
[Date of registration] 19.07.2001
[Number of appeal against examiner's decision of rejection]
[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報（A）**　(11)特許出願公開番号

特開平6－168145

(43)公開日　平成６年(1994)６月14日

(51)Int.Cl.[5]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　9/46	３４０　Ｆ	8120－5B		
12/00	５７２	9366－5B		

審査請求　未請求　請求項の数９（全 21 頁）

(21)出願番号　特願平3－352297

(22)出願日　平成３年(1991)12月13日

(31)優先権主張番号　特願平3－39269

(32)優先日　平３(1991)２月８日

(33)優先権主張国　日本（ＪＰ）

特許法第30条第１項適用申請有り　1990年12月18日～12月19日　社団法人トロン協会主催の「第７回トロンプロジェクト国際シンポジウム」において文書をもって発表

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地

(72)発明者　清水　剛
東京都小平市上水本町５丁目20番１号　株式会社日立製作所武蔵工場内

(72)発明者　竹山　寛
東京都小平市上水本町５丁目20番１号　株式会社日立製作所武蔵工場内

(72)発明者　小早川　学
東京都小平市上水本町５丁目20番１号　株式会社日立製作所武蔵工場内

(74)代理人　弁理士　玉村　静世

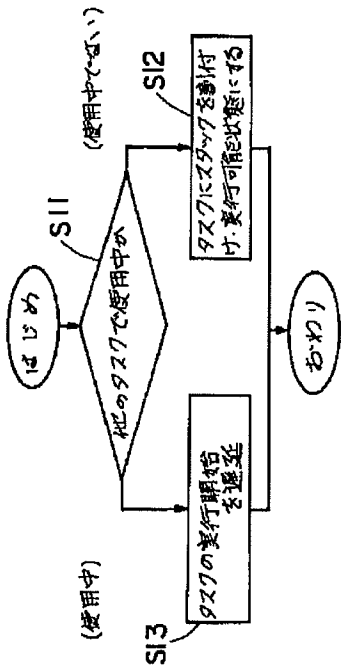
(54)【発明の名称】　スタック制御方式及びマイクロコンピュータ

(57)【要約】

【目的】　本発明の目的は、スタック領域をタスクに割り付ける処理が簡単であって、且つ、スタック領域のためのメモリ利用効率を向上させることができるマルチタスク処理システムにおけるスタック制御方式を提供することにある。

【構成】　複数のタスクに同一スタック領域を共有させ、個々の共有スタック領域に対しては、タスクの実行開始から終了までを単位として、前記共有スタック領域に対するタスクの割り付けを排他的に制御するものである。例えば、タスクの実行が要求されたとき、スタック領域を共有する他のタスクが当該共有スタック領域を使用しているか否かを判定し（Ｓ１１）、他のタスクが実行中でなければ当該タスクを実行可能にし（Ｓ１２）、他のタスクが実行中であれば当該タスクの実行可能な状態への遷移を遅延させる（Ｓ１３）ようにする。

【図１】



1

【特許請求の範囲】

【請求項1】　オペレーティングシステムの管理の下で複数のタスクを実行制御するマルチタスク処理システムにおいて、複数のタスクに同一スタック領域を共有させ、それらタスクの実行開始から終了までを単位として、前記共有スタック領域に対するタスクの割り付けを排他的に制御することを特徴とするスタック制御方式。

【請求項2】　前記排他的な制御は、タスクの実行が要求されたとき、スタック領域を共有する他のタスクが実行中であるか否かを判定する処理と、この判定結果により、実行中でなければ当該タスクを実行可能にする処理と、他のタスクが実行中であれば当該タスクの実行開始を遅延させる処理とを含むことを特徴とする請求項1記載のスタック制御方式。

【請求項3】　実行が要求されたタスクが割り付けられるべき共有スタック領域を、タスクに固有のスタック指示情報から求める処理を更に含むことを特徴とする請求項2記載のスタック制御方式。

【請求項4】　他のタスクが実行中であるか否かを判定する前記処理は、共有スタック領域に現在割り付けられているタスクが存在していることを指示するためのスタック管理変数を参照して行うものであることを特徴とする請求項2又は3記載のスタック制御方式。

【請求項5】　前記遅延させる処理は、共有スタック領域固有のスタック行列管理ポインタに実行開始が遅延されるタスクを繋ぐ処理であることを特徴とする請求項2乃至4の何れか1項記載のスタック制御方式。

【請求項6】　前記排他的な制御は更に、タスクの実行を終了したときに、そのタスクとスタック領域を共有していて実行開始が遅延されている他のタスクが存在するか否かを判定する処理と、これにより存在すると判定されたときに、実行開始が遅延されている単数若しくは複数のタスクの中から選ばれた一つのタスクを実行可能にさせる処理とを含むことを特徴とする請求項2記載のスタック制御方式。

【請求項7】　実行開始が遅延されている他のタスクが存在するか否かを判定する前記処理は、スタック行列管理ポインタの値を参照して、当該ポインタの値が他のタスクに繋がれていないことによって判定することを特徴とする請求項6記載のスタック制御方式。

【請求項8】　選ばれた一つのタスクを実行可能にさせる前記処理は、当該タスクを、実行可能なタスクの待ち行列を形成するための実行待ち行列管理ポインタに繋ぐ処理であることを特徴とする請求項6又は7記載のスタック制御方式。

【請求項9】　マルチタスク処理されるべき複数のタスクがスタック領域を共有したとき、タスクにそのスタック領域を、タスクの実行開始から終了までを単位として排他的に割り付け制御するための、スタック制御領域を確保する機能を持つオペレーティングシステムを搭載してなる1チップ型のマイクロコンピュータ。

2

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、オペレーティングシステム若しくはオペレーティングシステムプログラム（以下ＯＳとも記す）のメモリ管理方式に関し、特にＯＳの管理の下で中央処理装置が複数のタスク（ユーザープログラム）を実行制御するマルチタスク処理システムにおいて、作業用のスタック領域を各タスクに割り当てるためのスタック制御方式、並びに同制御方式を採用可能なシングルチップマイクロコンピュータに関する。

【0002】

【従来の技術】マルチタスク処理においては、システムの処理を独立して並列に処理可能な単位即ちタスクに分割して、これをオペレーティングシステム（ＯＳ）が管理しながらその実行を制御する。斯るマルチタスク制御ＯＳは、タスクからのシステムコール（タスクの実行要求）に対してマイクロプロセッサやその他の資源を優先度などに応じて当該タスクに割り当ててそのタスクの実行を制御する。ＯＳは、このときタスクにスタック領域を割り当てる。このスタック領域は、サブルーチンへのパラメータの引き渡しやサブルーチンからの戻り番地の保持並びにタスクの実行を中断したときのレジスタ情報の待避・復帰処理などに利用される。タスクに割り当てられるスタック領域が各タスクに夫々固有の領域とされると、登録されるタスクの数又は登録可能なタスクの数に比例してメモリの使用量が増大する。特に仮想記憶をサポートしていないデータ処理システムにおいてはスタック領域に使用されるメモリ容量の増大が顕著である。そこでスタック領域のためのメモリ使用量を削減するための技術として特開昭６３－８６０３５号がある。この特開昭６３－８６０３５号は、スタック領域を一括プール化し、必要に応じてそのプールから動的にスタック領域を切り出してタスクに割り付ける方式を開示している。

【0003】

【発明が解決しようとする課題】しかしながら、タスクの実行要求に応じてプール領域から動的にスタック領域を切り出してタスクに割り当てる方式においては、その動的な制御故にＯＳの処理が複雑化することがわかった。さらに動的にスタック領域を切り出し制御したとしてもスタック領域が不足しないようにそのプール領域の記憶容量を定める必要がある。すなわち、同時に実行されるタスクの数を予め計算した上で、プール領域の容量が決定される。したがって、スタック領域のためのメモリ利用効率が低下してしまうことがある。

【0004】特に仮想記憶を有しないシステムにおいては、タスクに割り付け可能なメモリ容量には物理的な制限があるので、スタック領域のサイズ（記憶容量）をできるだけ小さなサイズにすることが必要になる。特に比

3

較的規模の小さなデータ処理システム、たとえば、中央処理装置と周辺の入出力（Ｉ／Ｏ）機能やランダム・アクセス・メモリ（ＲＡＭ），リード・オンリ・メモリ（ＲＯＭ）などを１チップに搭載したシングルチップマイクロコンピュータによって制御部が構成されるようなデータ処理システムの場合には、限られた記憶容量しか持たない内蔵ＲＡＭを用いて複数のタスクを実行しなければならず、スタック領域のサイズを更に小さくする必要性があることを本発明者は見い出した。また、物理的に限られたメモリ領域の中で、タスクの実行要求毎に動的にスタック領域を切り出したり、あるいは夫々のタスクに固有のスタック領域を割り付ける方法は、実行可能な若しくは登録可能なタスク数に自ずから制限を及ぼすことにもなる。

【０００５】本発明の目的は、スタック領域をタスクに割り付ける処理が簡単であって、且つ、スタック領域のためのメモリ利用効率を向上させることができるマルチタスク処理システムにおけるスタック制御方式を提供することにある。

【０００６】本発明の別の目的は、スタック領域として割当可能なメモリ領域が限られたデータ処理システムにおいて、実行可能な若しくは登録可能なタスク数を比較的多くすることができるマルチタスク処理システムにおけるスタック制御方式を提供することにある。

【０００７】更に本発明はそのようなスタック制御方式を採用可能な１チップ型のマイクロコンピュータを提供することを目的とする。

【０００８】本発明の前記並びにその他の目的と新規な特徴は本明細書の記述及び添付図面から明らかになるであろう。

【０００９】

【課題を解決するための手段】本願において開示される発明のうち代表的なものの概要を簡単に説明すれば下記の通りである。

【００１０】本発明に関するオペレーティングシステムには、同一の優先度を有する複数のタスクに一つのスタック領域を割り当てる様なスタック管理機能が持たせられる。したがって、一つのスタック領域は同一優先度を持つ複数のタスクに対して共用される。但し、一つのスタック領域を共通利用する複数のタスクは、それらが同時に実行されないタスクとされ、かつ、それらの実行順序が予め決定可能なタスクとされる。すなわち、タスクＡ，Ｂ及びＣに一つのスタック領域を共用させる場合、各タスクＡ，Ｂ及びＣの実行順序は、タスクＡの実行終了後、タスクＢが実行され、タスクＢの実行終了後、タスクＣが実行されるような実行順序とされるのが良い。言い換えるならば、共用スタック領域は、排他的に実行中の一つのタスク（Ａ，Ｂ又はＣ）に割り付けられる。

【００１１】また、本発明に関するＯＳは、一つのタスクに対して一つのスタック領域を専用として割り当てることができる。たとえば、１０個のタスクをＯＳが管理するとき、ＯＳは、二つのタスクにそれぞれ専用のスタック領域を割り当て、ある同一の優先度を有する四つのタスクに一つの共用のスタック領域を割り当て、他の同一の優先度を有する四つのタスクに一つの共用のスタック領域を割り当てる。結果的に、全スタック領域の数は１０個のタスクがあるにもかかわらず四つとされ得る。

4

【００１２】

【作用】したがって、本発明に従えば、タスクの総数がｎ個（ｎは正の整数）であってもスタック領域の総数をｍ＜ｎ（ｍは正の整数）とすることができるので、スタック領域が割り当てられるべきランダム・アクセス・メモリの全メモリ容量に対する全スタック領域の割合を低減することができる。

【００１３】その結果、シングルチップマイクロコンピュータの様に内蔵ランダム・アクセス・メモリ（ＲＡＭ）の記憶容量が制限される場合であっても、スタック領域として利用される記憶容量は、上記ＲＡＭの全記憶容量のわずかとされ得るので、シングルチップコンピュータにマルチタスクのオペレーティングシステムプログラム（ＯＳ）を搭載することが可能となる。

【００１４】

【実施例】まず最初に本発明に従うオペレーティングシステムに関し説明する。

【００１５】図５に示されるように、マルチタスク制御用のオペレーティングシステム（ＯＳ）は複数のタスクＰ１乃至Ｐｉを並列的に管理してその実行を制御する。

【００１６】図６には、ＯＳがタスクを管理するとき同タスクが採り得るタスク状態遷移図の一例が示される。同図に示される遷移状態は、休止状態、実行可能状態、実行状態、及び待ち状態である。

【００１７】休止状態とは、タスクがまだ起動されていない状態即ちタスクの実行要求がまだされていない状態、或いはタスクの実行が終了された状態である。実行可能状態とは、タスクを実行可能な状態であるが、優先度の高い他のタスクなどが実行されているためプロセッサが割り付けられず待っている状態である。前記実行状態とは、タスクにプロセッサが割り付けられて現在実行中の状態である。前記待ち状態とは、タスクが、プロセッサ以外の資源の獲得を待っている、又は何らかの事象の発生を待っている状態である。

【００１８】本発明に従うＯＳは、複数のタスクに同一スタック領域を共有させるようなメモリ管理機能を含ませられている。本発明のＯＳは、例えば図４示されるように、優先度が同一レベルであって実行順序が相互に予め決定されるようなタスク群に同一のスタック領域を共有させるように、スタック領域の割り当てを制御する。図４では、タスクＰ２，Ｐ３，Ｐ４のスタック領域Ｂ、タスクＰ５，Ｐ６，Ｐ７のスタック領域Ｃ、タスクＰ８，Ｐ９，Ｐ１０のスタック領域Ｄが夫々共有スタック

5

領域とされる。

【００１９】尚、タスクＰ１，Ｐ１１には、それぞれスタック領域Ａ及びＥが専用スタック領域として割り当てられる。そして、本発明のＯＳは、個々の共有スタック領域に対しては、タスクの実行開始から終了までを単位として、前記共有スタック領域に対するタスクの割り付けを排他的に制御する。

【００２０】図１は、本発明のＯＳ内に含ませられる前記排他的な制御のプログラムフローチャートの例が示されている。すなわち、タスクの実行が要求されたとき、スタック領域を共有する他のタスクが実行中であるか否か（換言すれば当該共有スタック領域を他のタスクが使用しているか否か）が判定される（Ｓ１１）。その結果他のタスクが実行中でないと判定されれば、ＯＳは当該タスクを実行可能にする（Ｓ１２）。一方、他のタスクが実行中であると判定されれば、ＯＳは当該タスクの実行開始即ち実行可能な状態への遷移を遅延させる（Ｓ１３）。

【００２１】図３は、タスクの実行開始を遅延させる前記処理の一例を示している。たとえば、図４のようにスタック領域Ｂを利用してタスクＰ２が実行状態にあるときに，同スタック領域を共有する他のタスクＰ３，Ｐ４の実行が要求されると、当該共有スタック領域Ｂに固有のスタック行列管理ポインタ（後述するＢｐ）にそれらタスクＰ３，Ｐ４を繋いで待ち行列を形成するような処理を採用することができる。この待ち行列は、例えば、先入れ先出し（ＦＩＦＯ）のキューイングテーブルで構成される。

【００２２】実行要求されたタスクが割り付けられるべきスタック領域を容易に認識可能にする為、各タスクに対し、それが割り付けられるべきスタック領域を指示するスタック指示情報若しくはスタック指示データ（後述するＳＰＤ）が設けられる。他のタスクが実行中であるか否かを判定できるようにする為、共有スタック領域に現在割り付けられているタスクが存在しているか否かを指示するためのスタック管理変数（Ｂｄ）が共有スタック領域Ｂに対して設けられる。したがって、ＯＳはタスクが実行要求されると、そのタスクが割り付けられるスタック領域を認識するために、そのタスクのスタック指示情報をアクセスする。その結果、スタック指示情報が、例えば、共用スタック領域を指示している時、上記共有スタック領域の為に設けられたスタック管理変数をアクセスする。その結果、上記スタック領域に現在割り付けられているタスクが存在すると判定されれば、上記実行要求されているタスクは、待ち行列に入れられる。このことは、後述される図１２～図１４で詳細に説明する。

【００２３】共有スタック領域に対するタスクの前記排他的な割り付け制御の一貫として、実行開始が遅延されたタスクの取扱いを考慮する場合には、例えば図２に示

6

されるように、一つのタスクの実行を終了したとき、スタック領域を共有すると共に実行開始が遅延されているところの他のタスクが存在するか否かを判定する（Ｓ２１）。この判定の結果、実行開始が遅延されている他のタスクが存在する場合には、当該実行開始が遅延されている単数若しくは複数のタスクの中から選ばれた一つのタスクが実行可能状態にされる（Ｓ２２）。

【００２４】実行開始が遅延されている他のタスクが存在するか否かを容易に判定する為、図５に示されたスタック行列管理ポインタＢｐの値を参照して、当該ポインタの値が他のタスクを指しているか否かを判定する。

【００２５】当該実行開始が遅延されている複数のタスクの何れを選択するかのＯＳによる制御は、その待ち行列に含まれるタスクの優先度の高いものから、或いは待ち時間の長いものから、又はタスクの実行に要する時間が短いものから選択させたり、システムの要求に応じて適宜選択させることもできる。選ばれた一つのタスクを実行可能にさせる前記処理を比較的簡単にする為、実行可能なタスクを繋ぐためのレディータスク行列管理ポインタを設け、これにそのタスクを繋ぐようにされる。

【００２６】上述のスタック制御方式は、１チップ型のマイクロコンピュータのオペレーティングシステムとして有効である。その際マルチタスク処理されるべき複数のタスクにスタック領域を共有させるとき、一つのタスクの実行開始から終了までを単位としてそのタスクにそのスタック領域を排他的に割り付け制御するための共有スタック制御領域を確保する機能、すなわちスタック行列管理ポインタＢｐ、スタック指示情報ＳＰＤ、スタック管理変数Ｂｄを設け、それを管理する機能をオペレーティングシステムにもたせ、該オペレーティングシステムを搭載すればよい。

【００２７】この様にオペレーティングシステムを構成すれば下記の効果を得ることができる。複数のタスクに同一スタック領域を共有させ、タスクの実行開始から終了までを１単位として共有スタック領域に対するタスクの割り付けを排他的に待ち行列方式で制御するので、一括メモリプール化によるスタック領域の動的割り付け制御や仮想記憶を利用する制御に比べて、スタック領域をタスクに割り付ける処理プログラムを簡単にできる。同時に、共有スタック領域を設定するので、マルチタスク処理システムにおけるスタック領域のためのメモリ容量を低減可能であり、また、共有スタック領域を複数タスクで利用するので、メモリ利用効率を向上させることができる。このメモリ利用効率は、スタック領域に対する共有の度合いを増す程に向上する。

【００２８】マルチタスク処理システムにおけるスタック領域のためのメモリ利用効率の向上は、スタック領域として割当可能なメモリ領域が限られたデータ処理システムにおいても、実行可能な若しくは登録可能なタスク数をさほど制限しないように働く。

7

【0029】さらに、1チップ型のマイクロコンピュータ（シングルチップマイクロコンピュータ）のようにスタック領域として割当可能なメモリ領域が限られた中でも、管理すべきタスクを構成する各種ユーザープログラム若しくはアプリケーションプログラムに対してOSの柔軟な対応を可能にする。

【0030】以下、1チップ型のマイクロコンピュータ（シングルチップマイクロコンピュータ）を用いたデータ処理システムに本発明を適用した一実施例について説明する。

【0031】図7に示されるマイクロコンピュータ10は、特に制限されないが、オペレーティングシステムプログラム（OS）やタスク（ユーザプログラム若しくはアプリケーションプログラム）といった動作プログラムを実行する中央処理装置（CPU）11、上記OSやタスクといった動作プログラムなどを保有するROM（リード・オンリ・メモリ）12、前記中央処理装置11の作業領域やデータの一時記憶領域、例えばスタック領域などとして利用されるRAM（ランダム・アクセス・メモリ）13、入出力（I／O）インタフェース14、タイマ15を含み、それらはアドレスバス16、データバス17、及びコントロールバス18によって結合されている。上記マイクロコンピュータ10は、公知の半導体集積回路製造技術によって単結晶シリコンのような1個の半導体基板1上に形成されている。

【0032】上記タイマ15は、端子T1～T3を介して、外部に設けられたモータM1～M3の発生するパルス信号S1～S3を受けて、パルス信号の数をカウントする。前記I／Oインタフェース14は、キーボード51からの入力データを端子T4～T10を介して内部データバス17に供給し、内部データバス17上のデータを端子T14～Tnを介して表示装置（ディスプレイ）52に出力する。

【0033】図7のデータ処理システムの動作については後で詳述する。

【0034】上記OSは、タスク管理、タスク付属同期管理、同期・通信管理、時間管理、割込み管理などの機能を含む。以下に、その概要を示すと、（1）タスク管理（スケジューラを含む）は、CPU（Central Processing Unit）11をタスクに割り付ける順序やタスクの起動・終了など、タスクの状態を管理し、このときタスクは優先順位の高いものから順にCPUに割り付けられ、（2）タスク付属同期管理は、タスクの実行中断・再開など、タスクの基本的な同期処理を行い、（3）同期・通信管理は、イベントフラグ、セマフォ、メイルボックスの三つの機能があり、タスク間の同期・通信処理を行い、（4）時間管理は、時間の管理を行うと共に、タスクの実行制御のための時間監視を行い、（5）割込み管理は、割込み発生時に割込み処理ハンドラを起動し、割込み処理やタスクへの割込み発生の連絡を行う。

8

【0035】尚、本発明は、OSのタスク管理スタック領域割付け機能に関するものなので、上記（2）～（5）に示される管理内容に関しては詳細に説明されないが、当業者にとって容易に理解されるであろう。

【0036】前記中央処理装置11は、ROM12に記憶されたOSを核として各種タスクを所定の手順に従って実行制御する。CPU11は、実行すべきタスクの実行に際し、そのタスクを構成する動作プログラムをROM12から読み込んで解読するため、アドレスバス16に上記タスクの先頭アドレスに対応するアドレス信号を送出する。上記ROM12はアドレス信号に対応して、プログラムデータをデータバス17に送出する。CPU11は、上記プログラムデータを読み込んでこれを解読し、その解読結果に応じて各種演算制御や周辺回路の制御を行う。したがって、CPU11は、OSを実行している動作モード（スーパーバイザモード）とタスク（ユーザープログラム）を実行している動作モード（ユーザーモード）とを有している。上記OSを実行しているCPU11が、以下において説明される各種のタスク制御を実行していると見なされる。尚、図7には中央処理装置11が保有する各種レジスタとして、たとえば、16ビットのプログラムカウンタPCや8ビットのコンディションコードレジスタCCRのような制御レジスタ、データレジスタやアドレスレジスタなどとして使われる汎用レジスタ群Rn、及び現在実行中のタスクに割り付けられているスタック領域のトップアドレス（先頭アドレス）を指す16ビットのスタックポインタSTKPが代表的に図示されている。

【0037】汎用レジスタ群Rnは、たとえば、それぞれが16ビットとされる7本のレジスタR0～R6を含む。7本のレジスタR0～R6の内、たとえば、4本のレジスタR0～R3は、各タスクの文脈（プログラム）内でワークレジスタとして定義される。この様にする理由は、スタック領域として準備する記憶領域の容量を低減する為である。しかし、全ての汎用レジスタR0～R6をプログラム内でワークレジスタとして定義しても良い。このスタックされるべき汎用レジスタRnの本数は、システム構築時に特定することができる。

【0038】図8は、マイクロコンピュータ10のシングルチップモードでのアドレスマップを示している。ROM12は16kバイトとされ、16進アドレス（H′が付される）で表示するとH′0000からH′3FFFをしめる。ROMアドレスH′0000～H′003Dは、本発明と直接関係しないので詳細には説明されないが、インターラプトベクターテーブルを含む。H′003E～H′3FFFは、図9に示される様に、RAM13のアドレス領域を後述される図10の様に初期設定するためのデータテーブルを記憶する領域99、OSプログラムの記憶された領域100、タスクP1～P11

9

のプログラムが記憶された領域１０１～１１１、及びスタック指示データ（ＳＰＤ）テーブルの記憶された領域１１２を含む。このＳＰＤテーブルは、各タスクＰ１～Ｐ１１に対応して設けられるスタック指示データＳＰＤ１～ＳＰＤ１１を含む。これら各スタック指示データＳＰＤ１～ＳＰＤ１１は、各タスクが利用するスタック領域を示すための情報が記憶される。たとえば、各スタック指示データＳＰＤ１～ＳＰＤ１１は、対応するスタック領域のボトムアドレス、すなわち、アドレスマップ上で割り当てられる各スタック領域の最上位アドレス（終端アドレス）とされる。尚、ここで図８のアドレスマップ上において、アドレスの上位側とは、アドレスの大きな方向（Ｈ′ＦＦＦＦ方向）を示し、アドレスの下位側とはアドレスの小さな方向（Ｈ′０００方向）を示すと見なされる。さらに、スタック領域にあるデータがスタックされる場合、スタックポインタＳＴＫＰの値は、ボトムアドレス（スタック領域のアドレスマップ上での最上位アドレス）からスタックされるべきデータのバイト数を引いた値へと更新されるものとする。

【００３９】図４の様に、各スタック領域を各タスクに設定する場合、各スタック指示データＳＰＤ１～ＳＰＤ１１には夫々以下の様なデータがセットされる。

ＳＰＤ１	スタック領域Ａのボトムアドレス
ＳＰＤ２～４	スタック領域Ｂのボトムアドレス
ＳＰＤ５～７	スタック領域Ｃのボトムアドレス
ＳＰＤ８～１０	スタック領域Ｄのボトムアドレス
ＳＰＤ１１	スタック領域Ｅのボトムアドレス

【００４０】したがって、ＣＰＵ１１は、上記スタック指示データテーブル１１２内の所望のスタック指示データ記憶領域のアドレスをアドレスバス１６に出力し、リード状態を示すコントロール信号をコントロールバス１８に出力することによって、上記所望スタック指示データ記憶領域をアクセスする。そして、ＣＰＵ１１は、それに応答してＲＯＭ１３からデータバス１７に出力されたアドレスデータをリードすることによって、タスクがどのスタックに割り当てられているかを識別する。

【００４１】ＲＡＭ１３は、図８に示される様に、５１２バイトとされ、１６進アドレスで表示すると、Ｈ′ＦＤ８０～Ｈ′ＦＦ７Ｆに割り当てられる。ＲＡＭアドレスＨ′ＦＤ８０～Ｈ′ＦＦ７Ｆに割当てられる。ＲＡＭアドレスＨ′ＦＤ８０～Ｈ′ＦＦ７Ｆは、図１０に示される様に、１６バイトのＯＳスタック領域２０１、それぞれ１２バイトとされる各タスクのタスク管理テーブルＴＣＢ１ないしＴＣＢ１１を記憶する領域２０２～２１３、それぞれ２０バイトとされるスタック領域Ａないし Ｅ、それぞれ１バイトとされるスタック管理変数Ａｄないし Ｅｄを記憶する領域２１４ないし２１８、それぞれ２バイトとされるスタック行列管理ポインタＡｐないしＥｐを記憶する領域２１９ないし２２３、実行待ち行列管理ポインタＸｐを記憶する領域２４、及び２４７バイ

10

トのＣＰＵ１１の作業領域２２４を含む。

【００４２】尚、スタック管理変数Ａｄないし Ｅｄ及びスタック行列管理ポインタＡｐないしＥｐの領域２１４ないし２２３は共有スタック制御領域とされる。

【００４３】スタック管理変数領域２１４～２１８には対応するスタック領域が使用されているか否かを示す情報、又は対応スタック領域（Ａ～Ｅ）を使用しているタスクを特定するための情報、例えば、そのタスクの番号情報がＣＰＵ１１によって記述される。

【００４４】スタック行列管理ポインタＡｐないしＥｐは対応する共有スタック領域の空きを待っているタスクを待ち行列に繋ぐためのポインタとされる。

【００４５】図１０からも明かなようにスタック領域（Ａ）とこれに対応する共有スタック制御領域２１４、２１９は連続するアドレスにマッピングされている。従って、ＣＰＵ１１は、例えば、タスクＰ２の実行要求が他のタスク又はキーボード５１からされたとき、同タスクＰ２の実行に利用するスタック領域がＢであることを、上記内部アドレスバス１６にスタック指示データＳＰＤ２を示すアドレスを出力し、内部データバス１７に出力されるスタック指示データＳＰＤ２の内容から認識することができる。その共有スタック領域Ｂを他のタスクが使用しているか否かは、スタック指示データＳＰＤ２の内容が指しているアドレスの上位側次アドレスをＣＰＵ１１が参照（アクセス）すれば、そこに割り当てられているスタック管理変数領域２１５の保持情報Ｂｐから認識できる。さらに当該スタック領域Ｂを他のタスクが使用している場合に当該タスクをスタック待ち行列に繋げたり、あるいはスタック待ち行列に繋がっているタスクを認識したりする場合には、ＣＰＵ１１が、さらにその次アドレスのスタック行列管理ポインタＢｐの値を参照して行列を辿ったりすればよい。したがって、それらの手順は極めて簡単になる。１個のタスクに専用化されたスタック領域Ａ，Ｅのためのスタック管理変数Ａｄ，Ｅｄとスタック行列管理ポインタＡｐ，Ｅｐは当該スタック領域が１個のタスクに占有されていることを上記と同じ手順で認識させるために利用するものである。したがって、専用スタック領域であることを示す特別なフラグなどを設け、それをＣＰＵ１１が参照するような手順（プログラム）を採用すれば、専用スタック領域Ａ，Ｅに関してはスタック管理変数領域２１４，２１８やスタック行列管理ポインタ領域２１９，２２３をＲＡＭに割り当てなくてもよい。この様にすることによって、ＲＡＭ１３のメモリ容量を有効に利用できる。

【００４６】図８のアドレスマップに示される様に、１１２バイトの内蔵レジスタは、１６進アドレスで表示すると、Ｈ′ＦＦ９０からＨ′ＦＦＦＦに割り当てられる。これらのレジスタは、Ｉ／Ｏインタフェース１４内のレジスタ及び内蔵周辺回路であるタイマ１５内のレジスタに対応している。これらのレジスタに関しては、本

発明と直接関係しないので、詳細に説明されないが、当業者にとって容易に理解されるであろう。

【００４７】図１１にはタスク管理テーブルＴＣＢの一例が示される。タスク管理テーブルＴＣＢは、対応タスクの開始アドレス指定領域２０、タスクを待ち行列に繋ぐための各種ポインタ群２１、図６に示されるようなタスクの状態が記述される領域２２、及びタスクの優先度を記述する領域２３などを含む。例えば実行可能状態のタスクを繋げる待ち行列の先頭はＣＰＵ１１によって実行待ち行列管理ポインタ２４に記述される。その記述内容は対応タスクにおけるタスク管理テーブルのポインタ２１Ａのアドレスである。夫々のタスク管理テーブルにおけるポインタを利用して、ＣＰＵ１１は次々にタスクを繋げる様なことができる。同様にＣＰＵ１１による前記スタック待ち行列管理ポインタへの記述は、対応タスクにおけるタスク管理テーブルのポインタ２１Ｂのアドレスである。実際には一つのタスクがスタック待ち行列と実行待ち行列の双方に繋がれることはないので前記ポインタ２１Ａと２１Ｂは同一の領域が使用される。

【００４８】図１２にはタスク、スタック領域、及びスタック制御領域の関係が機能的に示されている。同図には共有スタック領域Ｂと専用スタック領域Ａが代表的に示されている。タスクＰ２，Ｐ３，Ｐ４の夫々が利用するところのスタック領域がＢであることは、夫々のスタック指示データＳＰＤ２，ＳＰＤ３，ＳＰＤ４が指しているＲＡＭ１３のアドレスによって認識される。タスクＰ１が利用するスタック領域がＡであることは該タスクに対応するスタック指示データＳＰＤ１が指しているＲＡＭ１３のアドレスによって認識される。同図においてタスクＰ２がスタック領域Ｂに割り付けられて実行状態にあるとき、スタック管理変数領域Ｂｄには当該スタック領域Ｂを利用しているタスクＰ２を示す情報（タスク番号）がＣＰＵ１１によって書き込まれている。このとき、スタック領域Ｂを共有する他のタスクＰ３の実行要求があると、ＣＰＵ１１はスタック行列管理ポインタＢｐにタスクＰ３に対応するタスク管理テーブルＴＣＢ３のポインタアドレスをスタック行列管理ポインタＢｐに記述して繋ぐ。続いてタスクＰ４の実行要求があると、ＣＰＵ１１は更にタスク管理テーブルＴＣＢ３のポインタにタスクＰ４に対応するタスク管理テーブルＴＣＢ４のポインタアドレスを記述してスタック待ち行列を形成する。一方タスクＰ１の実行が要求されていない状態においては、それ専用のスタック領域Ａに対応するスタック管理変数領域Ａｄは「空き」（即ち未使用）を意味する情報がＣＰＵ１１によって記述されており、また、スタック行列管理ポインタＡｐにもタスクを繋ぐための情報が記述されず「空き」の状態にされている。

【００４９】図１３にはタスクの実行要求に際してのスタック領域に対するタスク割付制御のフローチャートが示されている。この内容は、図１に対応された更に詳細なものに対応する。

【００５０】先ず、休止状態のタスクの中から一つのタスクに対して実行要求があると、ＣＰＵ１１は要求されたタスクのスタック指示データ（ＳＰＤ）から当該タスクが利用するスタック領域を求める（Ｓ３１）。例えば、図１２においてタスクＰ２の実行要求があると、ＣＰＵ１１はタスク２に対応するスタック指示データＳＰＤ２の指すＲＡＭ１３のアドレスによってスタック領域Ｂを認識する。

【００５１】次ぎに、ＣＰＵ１１はそのスタック領域Ｂを使って実行状態にある他のタスクが存在するか否かを判定するための情報を、スタック領域Ｂに対応するスタック管理変数領域Ｂｄから求める（Ｓ３２）。ＣＰＵ１１は該スタック管理変数領域の参照の為、前記スタック指示データを利用する。例えば、図１２のタスクＰ２に着目すると、ＣＰＵ１１はスタック指示データＳＰＤ２が指すＲＡＭ１３のアドレスの上位側次アドレスをアクセスすることによって、対応するスタック管理変数領域Ｂｄを参照する。

【００５２】前記スタック管理変数領域を参照した結果、その領域が「空き」の状態であれば、ＣＰＵ１１は当該スタック領域は未使用であると判定する。一方、当該領域にタスクの番号などの情報が保持されているとき、ＣＰＵ１１は他のタスクによって使用中であると判定する（Ｓ３３）。例えば、図１２に示されるように、スタック管理変数領域Ｂｄにタスクｐ２を意味する情報が記述されている状態において、そのスタック領域Ｂを共有するタスクＰ３の実行要求に従って、ＣＰＵ１１がスタック管理変数領域Ｂｄを参照した場合には、タスクＰ２が当該スタック領域Ｂに割り付けられていると判断する。また、図１２においてスタック管理変数領域Ａｄが「空き」の状態において、そのスタック領域Ａを利用しようとするタスクＰ１の実行要求に従って、ＣＰＵ１１がスタック管理変数領域Ａｄを参照した場合には、当該スタック領域Ａは未使用であると判定する。

【００５３】ステップＳ３３により使用中であると判定された場合には、今回実行要求されたタスクをスタック行列管理ポインタに繋ぐ（Ｓ３４）。そして斯るタスクを待ち状態にして実行を遅延させる（ステップＳ３５）。例えば、図１２に示されように、タスクＰ２が実行状態にあるとき、そのスタック領域Ｂを共有する他のタスクＰ３の実行要求があった場合、ＣＰＵ１１は、そのタスクＰ３に対応するタスク管理テーブルＴＣＢ３のポインタ（図１１のポインタ２１Ｂ）アドレスをスタック行列管理ポインタＢｐに記述して繋ぐ。続いてタスクＰ４の実行要求があると、更にＣＰＵ１１はタスク管理テーブルＴＣＢ３のポインタにタスクＰ４に対応するタスク管理テーブルＴＣＢ４のポインタアドレスを記述してスタック待ち行列を形成する。

【００５４】ステップＳ３３により未使用であると判定

された場合、ＣＰＵ１１は対応するスタック管理変数領域に該当タスク番号を記述する（Ｓ３６）。そして当該タスクが実行可能状態にされる（Ｓ３７）。

【００５５】特に、本実施例に従えば、スタック領域を共有するタスクとスタック領域を占有するタスクとが混在され、スタック領域ＡやＤのような専用スタック領域に対してもスタック管理変数が割り当てられている。そのため、タスクの実行要求に際してのスタック領域割り付け制御手順は、共有スタック領域に対しもまた専用スタック領域に対しても同じとすることができる。したがって、ＯＳのプログラム作成が簡単化され得る。

【００５６】図１４にはタスクの実行終了に際してのタスク割付制御の一例フローチャートが示される。この内容は、図２に対応し、実行開始が遅延されたタスクの取扱いとスタック領域の解放を考慮した更に詳細な内容を有する。

【００５７】先ず、タスクの実行が終了されるとき、ＣＰＵ１１は、そのタスクが占有していたスタック領域をスタック指示データから求める（Ｓ４１）。次ぎに、ＣＰＵ１１はそのスタック指示データが指すＲＡＭ１３のアドレスを元に対応するスタック行列管理ポインタを参照してスタック待ち行列にタスクが繋がれているかを判定するための情報を取得する（Ｓ４２）。そして、ＣＰＵ１１はその情報に基づいてスタック待ち行列を構成するタスク管理テーブルが存在するか否かを判定する（Ｓ４３）。

【００５８】ステップＳ４３の判定結果により該当するタスク管理テーブルが存在しない場合には、ＣＰＵ１１は対応するスタック管理変数に「空き」を意味する情報を書き込む（Ｓ４４）。これにより、当該スタック領域は、その後実行要求された所定のタスクに割り付け可能になる。

【００５９】ステップＳ４３の判定結果によりスタック待ち行列を構成するタスク管理テーブルが存在してる場合には、ＣＰＵ１１はスタック待ち行列に含まれる一つのタスク管理テーブルを当該スタック待ち行列からはずして当該スタック待ち行列を更新する（Ｓ４５）。どのタスク管理テーブルをはずすかは図２で説明した通り、スタック待ち行列に含まれるタスクの優先度、或いは待ち時間の長さ、又はタスクの実行に要する時間の長短などを考慮して決定される。そして、ＣＰＵ１１はスタック待ち行列からはずしたタスク管理テーブルに対応するタスクの番号などを対応するスタック管理変数領域に記述する（Ｓ４６）。さらに、ＣＰＵ１１はスタック待ち行列からはずしたタスク管理テーブルを今度は実行待ち行列に加えて当該タスク管理テーブルに対応するタスクを実行可能な状態にする（Ｓ４７）。

【００６０】上記スタック領域は、特に制限されないが、タスク内におけるサブルーチンへのパラメータ引き渡しやサブルーチンからの戻り番地の格納領域、さらには相対的に優先度の高いタスクの実行要求などによってそのときＣＰＵが割り付けられて実行状態にあったタスクが中断されるようなときに当該中断されるべきタスクの実行再開に必要な各種レジスタ情報の待避領域などとして利用される。このレジスタ情報の待避領域には、図７に示されるようなプログラムカウンタＰＣの値、コンディションコードレジスタＣＣＲの保有値、スタックポインタＳＴＫＰの値、並びに汎用レジスタ群Ｒｎの保有データなどが待避される。

【００６１】図１５にはタスクの実行状態におけるスタック領域の利用態様の一例が示される。例えばタスクＰ２の実行状態に着目すると、その過程において「ＫＥＩＳＡＮ」というサブルーチンに分岐するとき、当該タスクＰ２が割り付けられる共有スタック領域Ｂにはそのサブルーチンのためのパラメータ「Ｘ，Ｙが」引き渡されると共に、サブルーチンからの戻り番地例えば「１０００番地」が保持される。このときスタックポインタＳＴＫＰはスタック指示データＳＰＤ２の保有アドレスによってプリセットされ、これをスタック領域Ｂの開始番地として保有しており、スタック動作毎に順次デクリメントされていく。そして、サブルーチン「ＫＥＩＳＡＮ」の実行途上において、当該タスクＰ２よりも優先度の高いタスク（タスクＰ２とは異なるスタック領域を利用する）例えばタスクＰ１の実行要求によってそのサブルーチン「ＫＥＩＳＡＮ」を中断するときは、それまでに利用していた各種レジスタの情報例えばプログラムカウンタＰＣ、コンディションコードレジスタＣＣＲ、スタックポインタＳＴＫＰの値などをスタック領域Ｂに待避し、その後で当該タスクＰ２の実行を中断する。

【００６２】ここで、タスクの登録即ちタスクをＯＳに認識可能にする手法は、ＯＳとアプリケーションプログラムとを合成してＲＯＭへのロードモジュールを構成するようなプログラム構築時にＲＯＭ上のテーブルでＯＳに認識可能にするやり方と、タスクの生成や削除を行うシステムコールを用いる手法がある。このとき、上記説明では、どのタスクが何れのスタック領域を共有するかは、夫々のタスクと対を成すスタック指示データによってＯＳが認識する。したがって、上記共有スタック領域に対するタスクの割り付け制御をサポートするＯＳを利用したシステムにおいて、登録可能なタスク数が全スタック領域の容量に対して余裕があるようなアプリケーションプログラムを対象とする場合、必ずしも共有スタック領域を設定しなくてもよい場合があり、このようなとき全てのスタック領域をタスクに１対１対応する専用領域とすることもできる。全てのスタック領域が専用有領域である場合には、スタック管理変数領域やスタック行列管理ポインタのようなスタック制御領域を確保しなくてもよく、タスクの起動要求に応じて対応スタック領域を単に当該スタック領域に割り付ければよい。これを考慮すると、上記共有スタック領域に対するタスクの割り

付け制御をサポートするＯＳには、タスクの登録時に複数のタスクによるスタック領域共有状態を認識したときにスタック管理変数領域やスタック行列管理ポインタのようなスタック制御領域を確保する機能を与えておくことにより、システムに最適なスタック領域割り付け制御を行うことができる。

【００６３】図１６は、図７のデータ処理システムに適用されるＯＳとタスクとのタイムチャートの例を示している。尚、同図には理解を容易とするため、タスクＰ１ないしＰ５のみが示される。尚、これらタスクＰ１ないしＰ５のスタック領域及び優先度は図４に従うものとする。

【００６４】上記データ処理システムが起動されると、それに応答して初期化プログラム（図示されていない）、たとえば、タスクＰ１１が実行され、ＲＡＭ１３の図９に示される初期化データテーブル９９に従って、図１０の様にＲＡＭ１３のアドレスが初期化される。上記初期化プログラムの最後プログラム語には、例えばタスクＰ１を実行要求するための命令ｓｔｒ＿ｔｓｋＰ１が入れられており、上記ＯＳは、それの命令に基づいて、タスクＰ１を実行可能状態とする。すなわち、タスクＰ１内の命令ｓｔｒ＿ｔｓｋはシステムコール命令すなわち、ＯＳ内のタスクを実行状態とするためのサブルーチンタスク起動（ｓｔａｒｔ－ｔａｓｋ）を呼びだす命令とされる。尚、図１３に示されるプログラムフローチャートがこのサブルーチンに対応する。

【００６５】タスクＰ１は上記データ処理システムのメインプログラムと見なされる。モータＭ１～Ｍ３を所定の速度で回転させる様なプログラム及び速度調整の為のプログラムを含む。図示される様に、タスクＰ１の内部プログラムには、タスクＰ２，Ｐ３，Ｐ４を実行要求する命令ｓｔｒ＿ｔｓｋＰ２，ｓｔｒ＿ｔｓｋＰ３，ｓｔｒ＿ｔｓｋＰ４が含まれる。したがって、タスクＰ１がタスクＰ２，Ｐ３及びＰ４を次々と実行要求する。その結果、システムコール命令ｓｔｒ＿ｔｓｋＰ２に応答し、ＯＳ内のサブルーチンｓｔａｒｔ－ｔａｓｋが起動されタスクＰ２に共有スタック領域Ｂを割りつけて実行可能状態とする。そして次のシステムコール命令ｓｔｒ＿ｔｓｋＰ３に応答して再度サブルーチンスタートタスク（ｓｔａｒｔ＿ｔａｓｋ）が起動される。上記タスクＰ３は上記タスクＰ２とスタック領域Ｂを共有しかつ、既にスタック領域Ｂにタスクが割り当てられているので、タスクＰ３はスタック待ち行列につながれることになる。同様に、システムコール命令ｓｔｒ＿ｔｓｋＰ４に応答して再度サブルーチンスタートタスク（ｓｔａｒｔ＿ｔａｓｋ）が起動されるが、前記タスクＰ３同様にそのタスクＰ４もスタック領域Ｂを共有するため、タスクＰ４もまたスタック待ち行列につながれることになる。したがってスタック待ち行列は、タスクＰ３とそれにつづくタスクＰ４とがつながれることになる。この時同様に実行待ち行列にもタスクＰ３とタスクＰ４がつながれることになる。

【００６６】その後、タスクＰ１内のシステムコール命令ｓｔｒ＿ｔｓｋＰ１によって、タスクＰ１は自らを図６の待ち状態とする。すなわち命令ｓｔｐ＿ｔｓｋＰ１はＯＳ内のサブルーチンスリープタスク（ｓｌｅｅｐ＿ｔａｓｋ）を呼びだす命令とされる。上記サブルーチンスリープタスクは、それが起動されると、タスク管理テーブルＴＣＢのタスクの状態を記憶する領域２２のデータを待ち状態を示すデータとし、対応するタスクを待ち状態とする。

【００６７】次に、実行可能状態とされているタスクＰ２が実行される。タスクＰ２はたとえば、モータＭ１，Ｍ２，Ｍ３から出力される各パルス信号Ｓ１～Ｓ３をタイマ１５で計測するプログラムとされる。

【００６８】タスクＰ２のプログラム語の最後にはシステムコール命令Ｅｘｔ＿ｔｓｋＰ２が書かれている。このシステムコール命令Ｅｘｔ＿ｔｓｋＰ２はＯＳのサブルーチンタスク終了（Ｅｘｉｔ－ｔａｓｋ）を呼び出す命令とされ、それによって図１４のプログラムフローチャートが実行される。したがって、システムコール命令Ｅｘｔ＿ｔｓｋＰ２が実行されるとタスクＰ２は終了される（休止状態とされる）ことになり、スタック行列管理ポインタが調べられる。

【００６９】この実施例の場合、タスクＰ１からタスクＰ２ないしＰ４の実行要求が出されているので、ＯＳはタスクＰ２を待ち行列からはずし、タスクＰ３を実行可能状態とする。そして、タスクＰ３が実行される。タスクＰ３は、タスクＰ２で得られた計測データをＣＰＵ１１で演算し、各モータＭ１ないしＭ３の回転速度データを求めるプログラムとされる。

【００７０】次に上記タスクＰ３の実行はシステムコール命令Ｅｘｔ＿ｔｓｋＰ３にしたがって終了される（休止状態にされる）と、ＯＳは、タスクＰ３を待ち行列からはずし、タスクＰ４を実行可能状態とする。そして、タスクＰ４が実行される。

【００７１】タスクＰ４は、上記タスクＰ３で得られた速度データを表示装置５２に表示するプログラムとされる。タスクＰ４の実行は、システムコール命令Ｅｘｔ＿ｔｓｋＰ４に従って、終了される（休止状態とされる）。タスクＰ４のプログラムにおいて、上記システムコール命令Ｅｘｔ＿ｔｓｋＰ４の前には、システムコール命令ｗｕｐ＿ｔｓｋＰ１が記入されている。このシステムコール命令ｗｕｐ＿ｔｓｋはＯＳ内のサブルーチンウェイクアップタスク（ｗａｋｅｕｐ－ｔａｓｋ）を呼びだす命令とされ、それによって待ち状態とされているタスクを実行可能状態とする。したがって、図１６において、タスクＰ１が待ち状態とされているタスクであり、上記システムコール命令ｗｕｐ＿ｔｓｋＰ１によって、タスクＰ１が実行可能状態に復帰される。すなわ

ち、タスクＰ１のタスク管理テーブルＴＣＢ１内のタスクの状態を記憶する領域２２のデータが、待ち状態を示すデータから実行状態を示すデータに書き変えられる。そして、タスクＰ１が実行される。これによってタスクＰ３の結合に基づいて、モータの回転速度を所定値に設定する為のタスクＰ１内のプログラムが実行されようとする。

【００７２】図１６では、その後キーボード５１の入力によって、システムコール命令ｉｓｔｒ＿ｔｓｋＰ５が発生されている。このシステムコール命令ｉｓｔｒ＿ｔｓｋは、外部キー入力などのデータに応答して発せられる様な命令で、非タスク部からのタスク実行要求命令とされる。この命令ｉｓｔｒ＿ｔｓｋは、ＯＳ内のサブルーチン非タスク部の為のタスク起動（ｓｔａｒｔ　ｔａｓｋ　ｆｏｒ　ｔａｓｋ－ｉｎｄｅｐｅｎｄｅｎｔ　ｐｏｒｔｉｏｎ）を呼びだす命令である。そのサブルーチンによって、タスクＰ１の実行は中断され、待ち状態とされ、タスクＰ５が実行される。したがって、タスクに依存しない外部入力によってもタスクの実行要求を発行することができる。前記タスクＰ５は、たとえば、モータＭ１～Ｍ３の回転を停止させる様な出力信号０１～０３をシングルチップマイコン１０から出力させるプログラムとされる。この様にしてＯＳはタスクＰ１ないしＰ５の実行を制御する。

【００７３】上記実施例によれば以下の作用効果を得るものである。

【００７４】（１）複数のタスクに同一スタック領域を共有させ、タスクの実行開始から終了までを１単位として共有スタック領域に対するタスクの割り付けを排他的に制御することにより、一括メモリプール化によるスタック領域の動的割り付け制御や仮想記憶を利用する制御に比べて、スタック領域をタスクに割り付ける処理を簡単にすることができる。これと同時に、マルチタスク処理システムにおけるスタック領域のためのメモリ利用効率を向上させることができる。このメモリ利用効率は、スタック領域に対する共有度合いを増す程向上する。スタック領域に対する共有度合いを増すには、一連の実行順序が一義的に決定される性質のタスクを多く採用すればよく、換言すれば、往々にして動作パターンが限られるような機器の制御に最適である。

【００７５】（２）上記により、マルチタスク処理システムにおけるスタック領域のためのメモリ利用効率が向上すれば、スタック領域として割当可能なメモリ領域が限られた中でも、実行可能な若しくは登録可能なタスク数が制限される事態を防止することができる。したがって、１チップ型のマイクロコンピュータのようにスタック領域として割当可能なメモリ領域が限られた中でも、タスクを構成する各種ユーザープログラム若しくはアプリケーションプログラムに対してＯＳを柔軟に対応させることができる。

【００７６】（３）共有スタック領域と専用スタック領域が混在される場合に、その専用スタック領域に対してもスタック管理変数領域やスタック行列管理ポインタのようなスタック制御領域を確保することにより、双方のスタック領域に対するタスクの割り付け制御に図１３及び図１４で説明したような同一手順を採用することができ、スタック割り付け制御を一層簡素化することができる。

【００７７】（４）従来のスタック領域一括プール化によるスタック領域動的割り付け制御に対し、スタック待ち行列によりどのタスクがどのタスクによって実行遅延されているか否かが明らかになり、これをその他適宜のプロセスシーケンスにも利用することが可能になる。

【００７８】以上本発明者によってなされた発明を実施例に基づいて具体的に説明したが、本発明はそれに限定されるものではなく、その要旨を逸脱しない範囲において種々変更可能であることは言うまでもない。

【００７９】例えば上記実施例では、共有スタック領域に対するタスクの割り付け制御により実行を遅延させるべきタスクをポインタで繋いでスタック待ち行列を形成して管理する場合について説明したが、実行を遅延させるべきタスクの番号情報若しくはそのタスク管理テーブルアドレスなどを特定の領域に順番に書き込んで管理するような配列管理などの手法を採用することもできる。但し斯る手法は、スタック待ち行列に比べてメモリ使用量が僅かながら増えると予想される。

【００８０】また、ＯＳのリアルタイム性若しくはタスクの緊急的な実行を考慮する場合には、任意のタスクによって共通利用可能なスタック領域を予め確保しておき、非同期で発生する事象に対処しなければならないようなタスクの優先的な実行にも対応できるようにすればよい。

【００８１】また、共有スタック領域に対するタスクの割り付け制御を実現するに当たり、どのタスクがどのスタック領域を共有するかをＯＳに認識可能にするための手法は上記実施例の説明に限定されず、明示的若しくは暗黙的にタスクとスタック領域とを論理的に結合する以下の手法がある。

（ａ）スタック領域を共有するタスク或いはタスク群をテーブルを用いてＯＳに認識させる。

（ｂ）ロードモジュールの生成若しくはタスクの生成などのタスク登録時に、登録されるタスクが使用するスタック領域を指定させ、その際他のタスクと同じスタック領域が指定されているときは、その指定されたスタック領域を共有させる。

（ｃ）同一優先度を持つタスクには同一スタック領域を自動的に共有させる。

（ｄ）タスク登録時に他のタスクを指定させ、指定されたタスクが使用するスタック領域を当該登録されるべきタスクにも共有させる。

（e）タスクを登録したタスクのためのスタック領域を当該登録されるべきタスクに共有させる。

【0082】以上の説明では主として本発明者によってなされた発明をその背景となった利用分野である1チップ型のマイクロコンピュータに適用した場合について説明したが、本発明はそれに限定されず、種々のコンピュータシステムに広くて適用することができる。

【0083】本発明は、少なくともOSによってタスクを管理してその実行を制御するマルチタスク制御を条件とするものに広く適用することができる。

【0084】

【発明の効果】本願において開示される発明のうち代表的なものによって得られる効果を簡単に説明すれば下記の通りである。

【0085】すなわち、1個のスタック領域を複数のタスクが共有することにより、スタック領域のためのメモリ利用効率を向上させることができるという効果がある。

【0086】これにより、従来メモリ容量不足によって満足のいくマルチタスク処理が行えなかったシステムにおいてもOSによるマルチタスク処理の実現が可能になるという効果がある。

【0087】それと共に、スタック領域として割当可能なメモリ領域が限られた中でも、実行可能な若しくは登録可能なタスク数の制限を緩和することができるという効果がある。

【0088】更に、それらタスクの実行開始から終了までを1単位として共有スタック領域に対するタスクの割り付けを排他的に制御することにより、複雑なメモリ制御が不要になり、しかも、プログラムサイズも小さくなり、処理の高速化も可能になる。

【0089】さらに、1チップ型のマイクロコンピュータのようにスタック領域として割当可能なメモリ領域が限られた中でも、管理すべきタスクを構成する各種ユーザープログラム若しくはアプリケーションプログラムに対してOSの柔軟な対応を可能にする。

【図面の簡単な説明】

【図1】図1は本発明に係るスタック制御方式をタスクの実行要求に際しての処理に着目した原理的な一例フローチャートである。

【図2】図2は本発明に係るスタック制御方式をタスクの実行終了に際しての処理に着目した原理的な一例フローチャートである。

【図3】図3は図1の処理によってタスクの実行開始が遅延された状態を示す一例説明図である。

【図4】図4はタスクによるスタック領域の共有関係を示す一例説明図である。

【図5】図5はマルチタスク制御OSとタスクとの一例関係説明図である。

【図6】図6はマルチタスク制御OSがタスクを管理するときの一例状態遷移図である。

【図7】図7は本発明に係る1チップマイクロコンピュータの一実施例ブロック図である。

【図8】図8は図7に示されるマイクロコンピュータの一例アドレスマップである。

【図9】図9は図7に示されるマイクロコンピュータの内蔵ROMにおける一部のアドレスマップである。

【図10】図10は図7に示されるマイクロコンピュータの内蔵RAMのアドレスマップである。

【図11】図11はタスク管理テーブルの一例説明図である。

【図12】図12はタスク、スタック領域、及びスタック制御領域の一例関係説明図である。

【図13】図13はタスクの実行要求に際してのスタック領域に対するタスク割付制御の一例フローチャートである。

【図14】図14は実行開始が遅延されたタスクの取扱いとスタック領域の解放を考慮してタスクの実行終了に着目したタスク割付制御の一例フローチャートである。

【図15】図15はタスクの実行状態におけるスタック領域の一例説明図である。

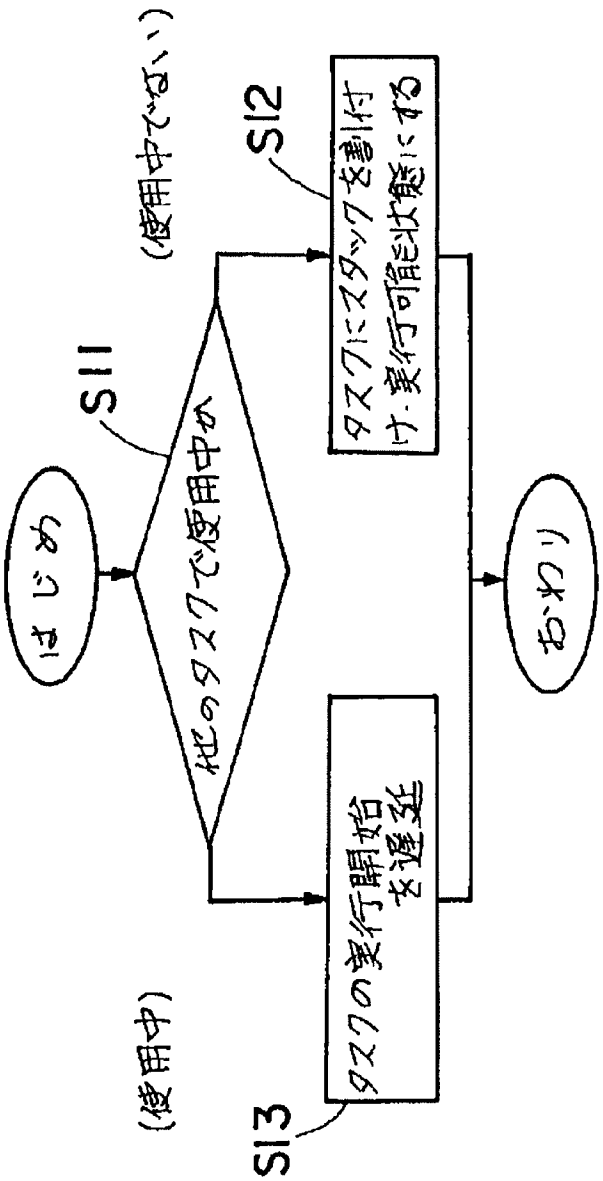
【図16】図16はCPUのOS実行状態とタスク実行状態のタイムチャートを示す一例説明図である。

【符号の説明】

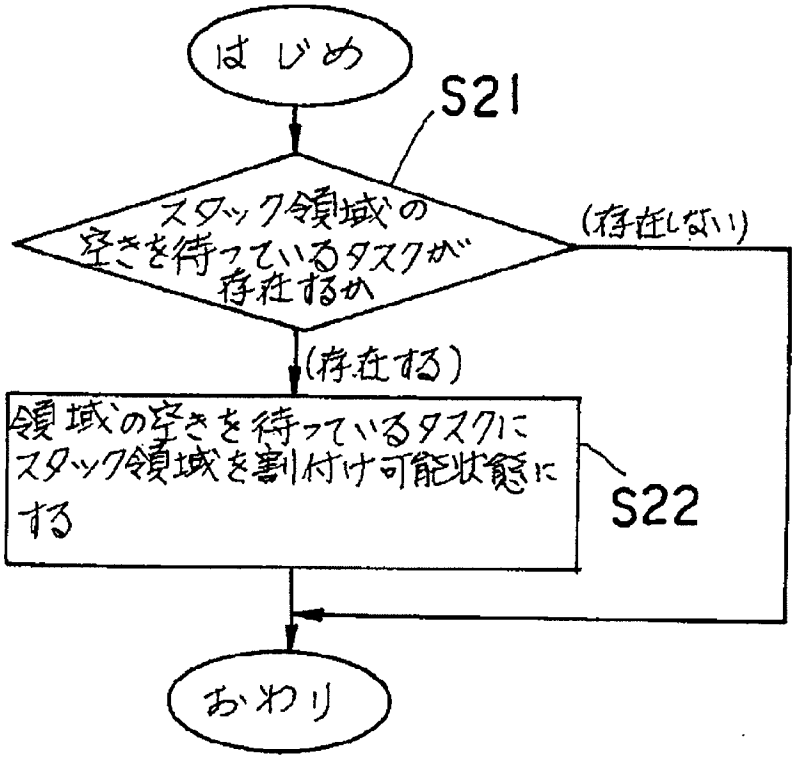
OS　オペレーティングシステム
P1乃至P11　タスク
A乃至E　スタック領域
10　マイクロコンピュータ
11　中央処理装置
12　ROM
13　RAM
14　I／Oインタフェース
15　タイマ
SPD1乃至SPDi　スタック指示データ
TCB1乃至TCBi　タスク管理テーブル
Ad乃至Ed　スタック管理変数
Ap乃至Ep　スタック行列管理ポインタ
21　ポインタ群
24　実行待ち行列管理ポインタ

【図1】

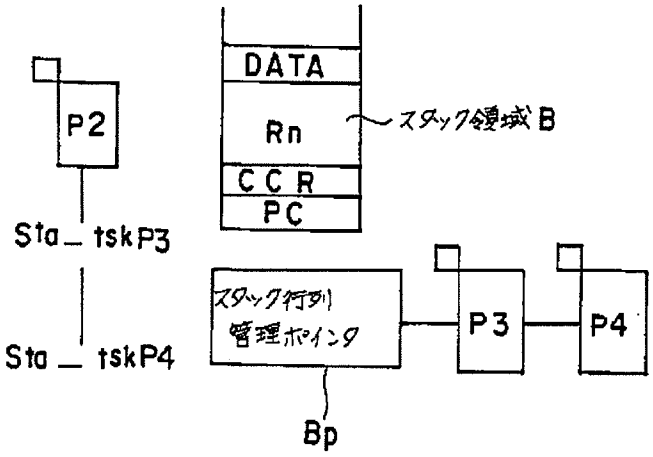
【 図 1 】


はじめ
S11
他のタスクで使用中か
(使用中でない)
S12
タスクにスタックを割付け・実行可能状態にする
(使用中)
S13
タスクの実行開始を遅延
おわり

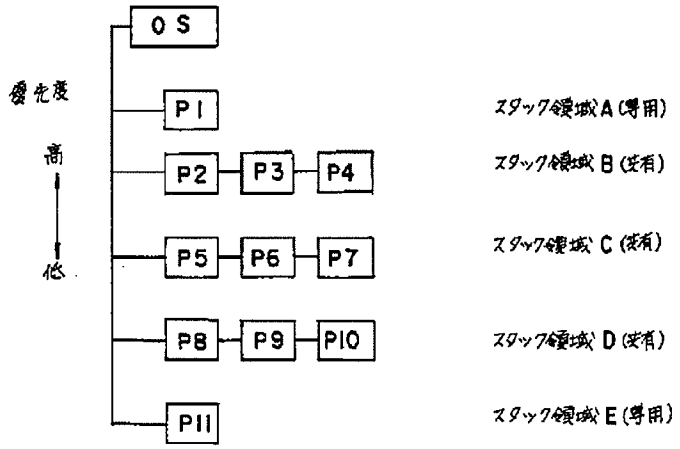
【図2】


はじめ
S21
スタック領域の空きを待っているタスクが存在するか
(存在しない)
(存在する)
領域の空きを待っているタスクにスタック領域を割付け可能状態にする
S22
おわり

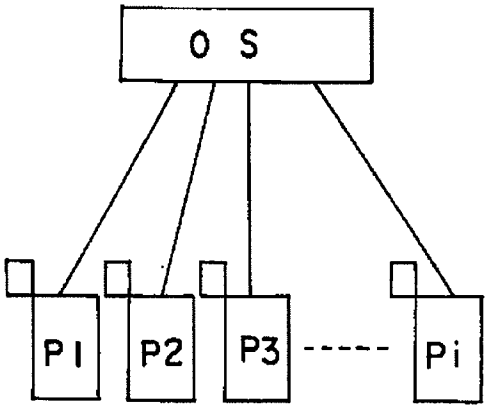
【図3】


DATA
Rn
スタック領域B
CCR
PC
P2
Sta_tskP3
Sta_tskP4
スタック行列管理ポインタ
P3
P4
Bp

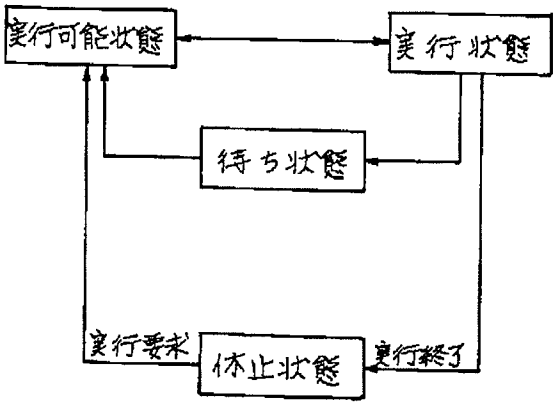
【図4】


OS
優先度
高
低
P1
P2
P3
P4
P5
P6
P7
P8
P9
P10
P11
スタック領域A(専用)
スタック領域B(共有)
スタック領域C(共有)
スタック領域D(共有)
スタック領域E(専用)

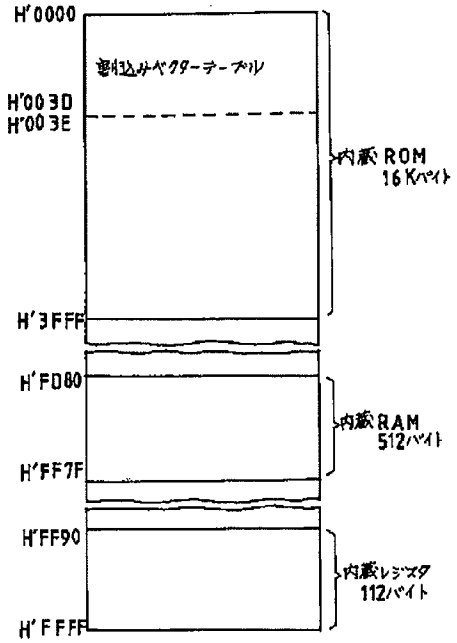
【図5】


OS
P1
P2
P3
Pi

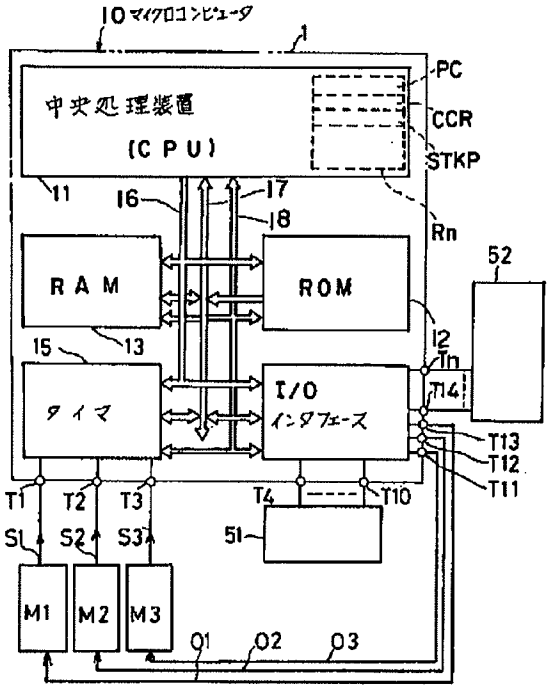
【図6】


実行可能状態
実行状態
待ち状態
実行要求
休止状態
実行終了

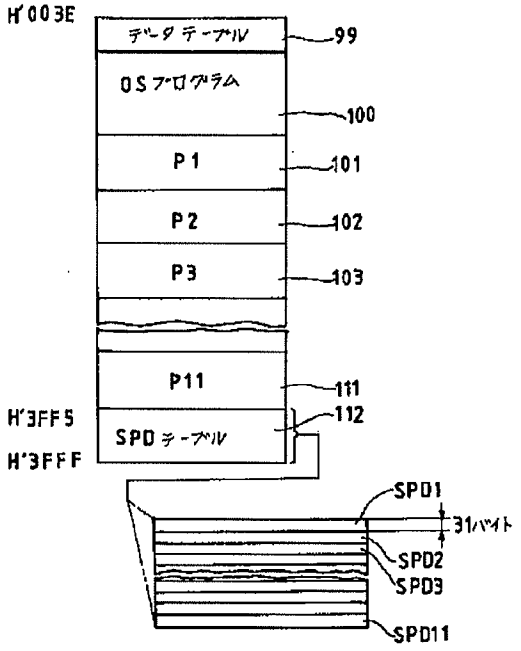
【図8】


H'0000
割込みベクターテーブル
H'003D
H'003E
内蔵ROM
16Kバイト
H'3FFF
H'FD80
内蔵RAM
512バイト
H'FF7F
H'FF90
内蔵レジスタ
112バイト
H'FFFF

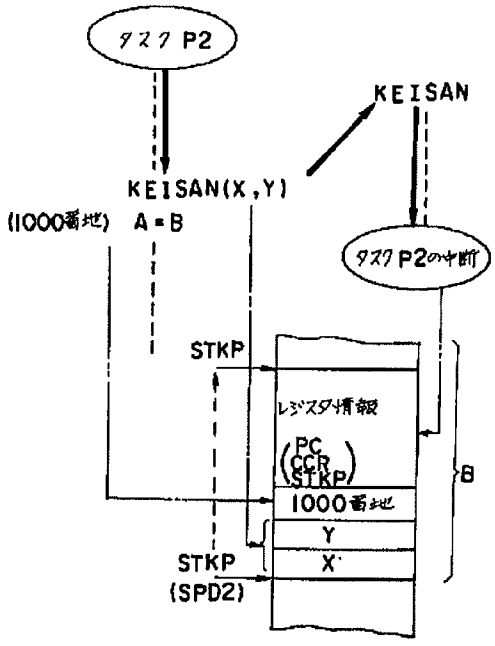
【図7】


10マイクロコンピュータ
1
中央処理装置
(CPU)
PC
CCR
STKP
Rn
11
16
17
18
RAM
ROM
52
12
Tn
15
13
タイマ
I/O
インタフェース
T14
T13
T12
T11
T1
T2
T3
T4
T10
S1
S2
S3
51
M1
M2
M3
O1
O2
O3

【図9】

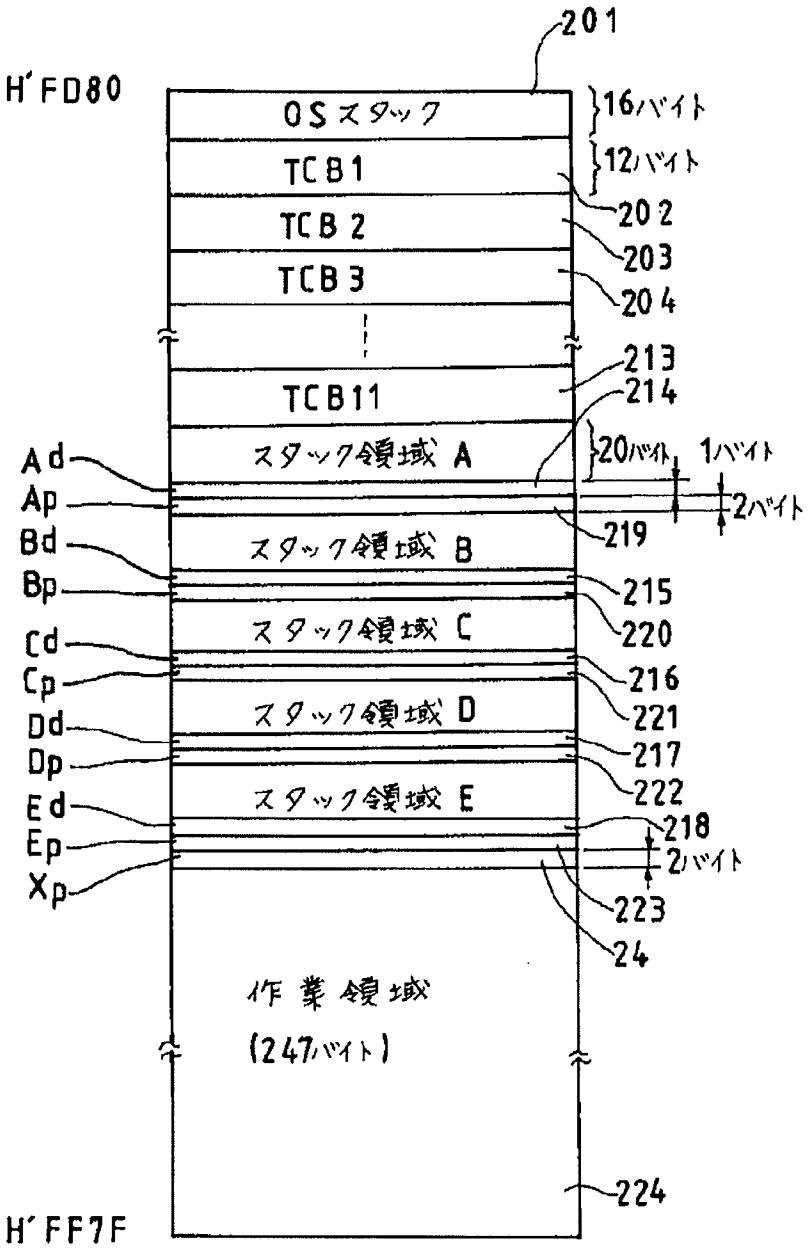

H'003E
データテーブル
99
OSプログラム
100
P1
101
P2
102
P3
103
P11
111
H'3FF5
SPDテーブル
112
H'3FFF
SPD1
31バイト
SPD2
SPD3
SPD11

【図15】


タスクP2
KEISAN
KEISAN(X,Y)
(1000番地)　A=B
タスクP2の中断
STKP
レジスタ情報
PC
CCR
STKP
1000番地
Y
X
8
STKP
(SPD2)

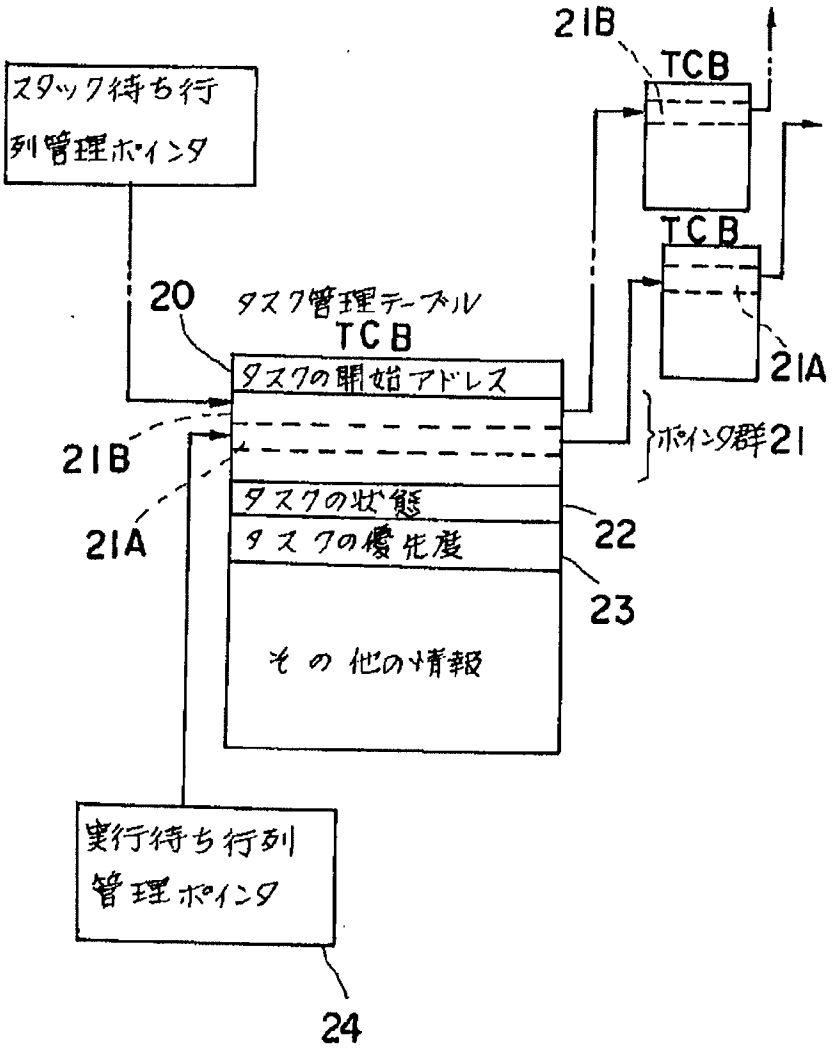
【図10】

【図10】


201
H'FD80
OSスタック
16バイト
TCB1
12バイト
202
TCB2
203
TCB3
204
213
TCB11
214
スタック領域 A
20バイト
1バイト
2バイト
Ad
Ap
219
Bd
スタック領域 B
Bp
215
220
Cd
スタック領域 C
Cp
216
221
Dd
スタック領域 D
Dp
217
222
Ed
スタック領域 E
Ep
218
2バイト
Xp
223
24
作業領域
(247バイト)
224
H'FF7F

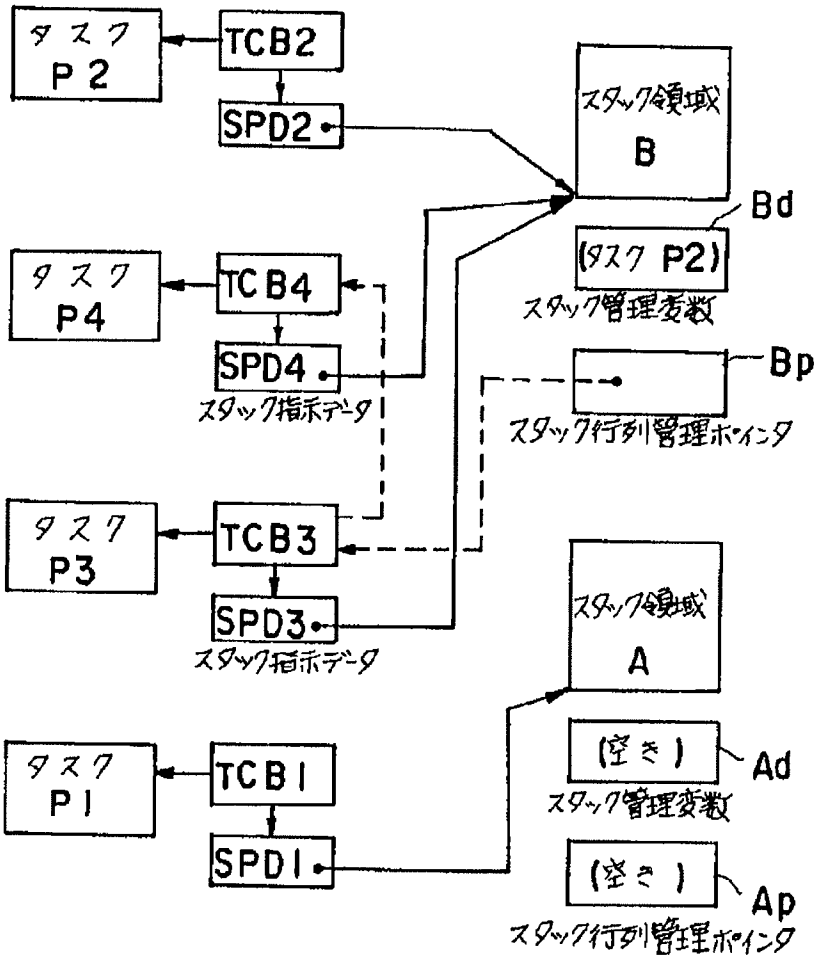
【図11】

【図11】


スタック待ち行列管理ポインタ
20
タスク管理テーブル
TCB
タスクの開始アドレス
21B
21A
ポインタ群21
タスクの状態
タスクの優先度
その他の情報
22
23
実行待ち行列管理ポインタ
24
21B
TCB
TCB
21A

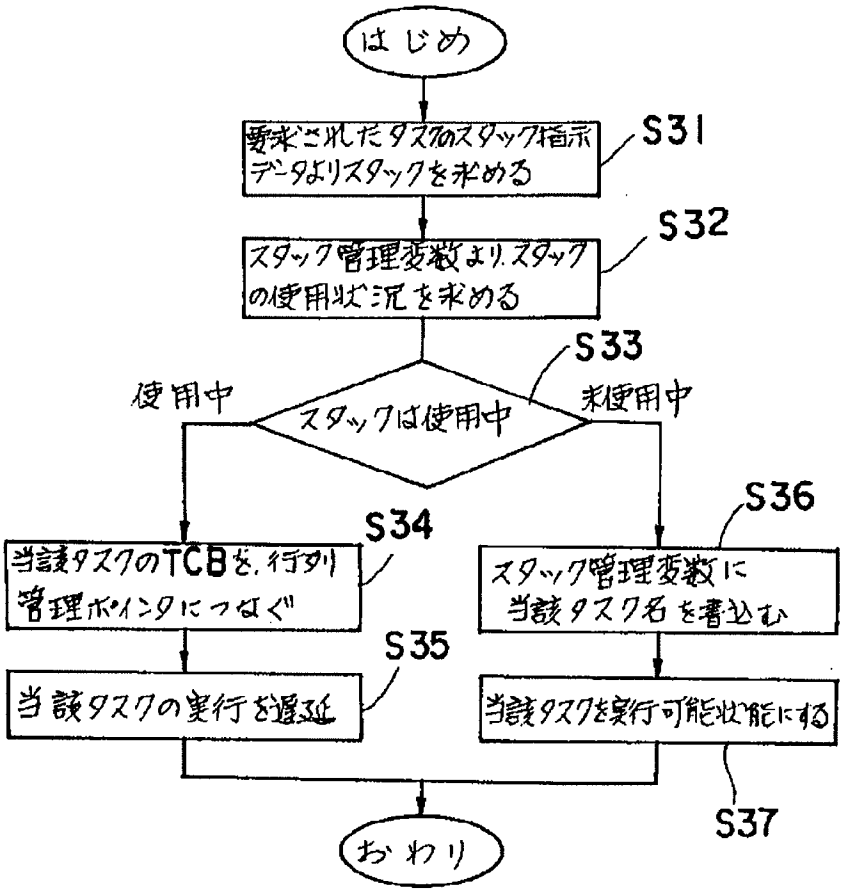
【図12】

【図12】


タスク P2
TCB2
SPD2
スタック領域 B
Bd
(タスク P2)
スタック管理変数
タスク P4
TCB4
SPD4
スタック指示データ
Bp
スタック行列管理ポインタ
タスク P3
TCB3
SPD3
スタック指示データ
スタック領域 A
タスク P1
TCB1
SPD1
(空き)
Ad
スタック管理変数
(空き)
Ap
スタック行列管理ポインタ

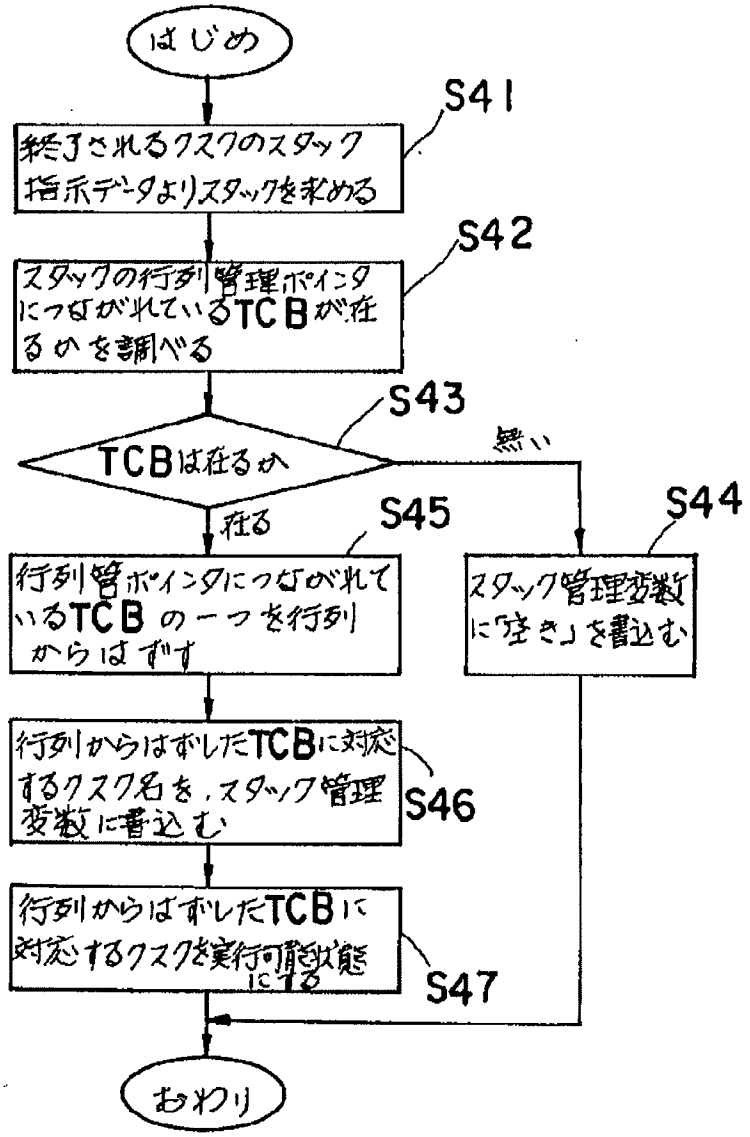
【図13】

【図13】


はじめ
要求された タスクの スタック指示 データより スタックを求める
S31
スタック管理変数より スタック の使用状況を求める
S32
S33
スタックは使用中
使用中
未使用中
当該タスクのTCBを、行列 管理ポインタにつなぐ
S34
当該タスクの実行を遅延
S35
スタック管理変数に 当該タスク名を書込む
S36
当該タスクを実行可能状態にする
S37
おわり

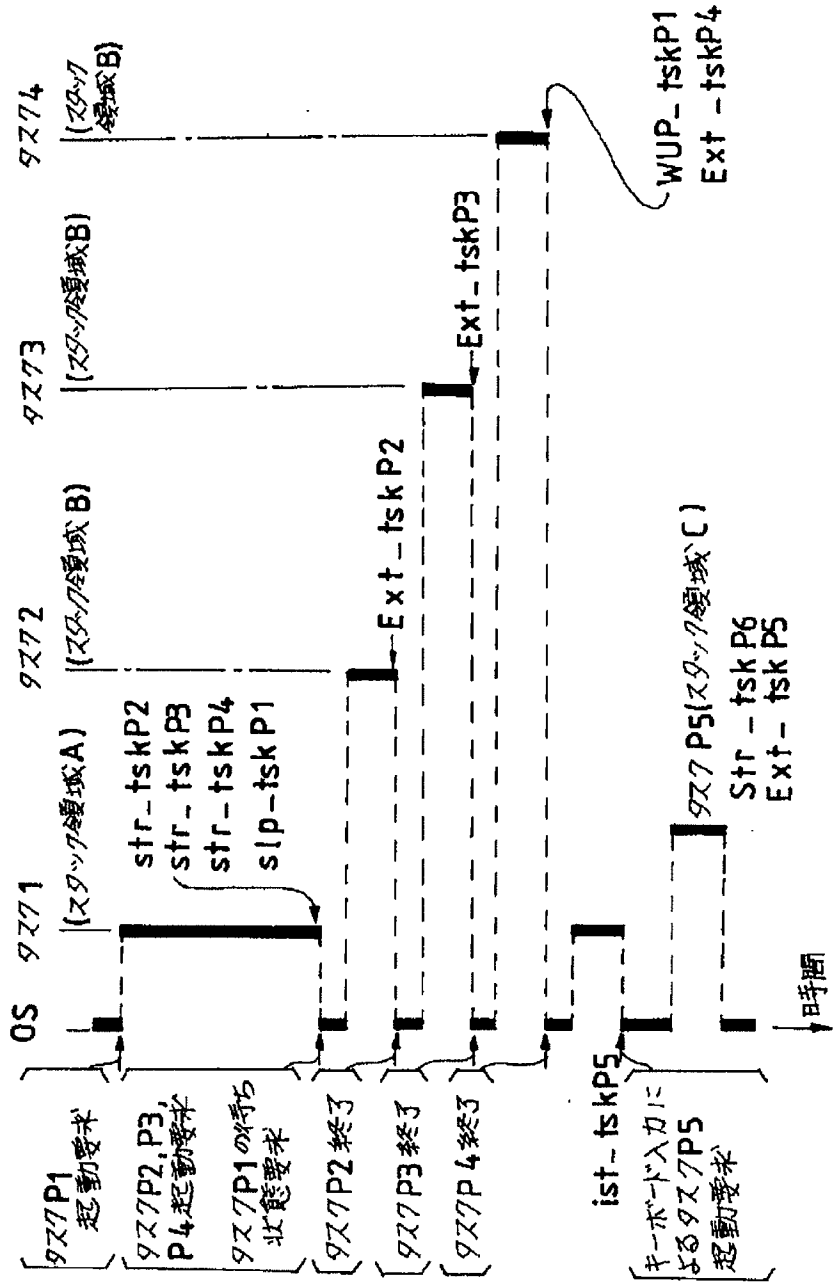
【図14】

【図14】


はじめ
S41
終了されるタスクのスタック指示データよりスタックを求める
S42
スタックの行列管理ポインタにつながれているTCBが在るかを調べる
S43
TCBは在るか
無い
在る
S44
スタック管理変数に「空き」を書込む
S45
行列管ポインタにつながれているTCBの一つを行列からはずす
S46
行列からはずしたTCBに対応するタスク名を，スタック管理変数に書込む
S47
行列からはずしたTCBに対応するタスクを実行可能状態にする
おわり

【図16】

【図16】


OS
タスク1
タスク2
タスク3
タスク4
(スタック領域A)
(スタック領域B)
(スタック領域B)
(スタック領域B)
str_tskP2
str_tskP3
str_tskP4
slp-tskP1
Ext_tskP2
Ext_tskP3
WUP_tskP1
Ext_tskP4
タスクP1 起動要求
タスクP2, P3, P4起動要求
タスクP1の待ち状態要求
タスクP2終了
タスクP3終了
タスクP4終了
ist_tskP5
キーボード入力によるタスクP5起動要求
タスクP5(スタック領域C)
Str_tskP6
Ext_tskP5
時間